Repetition: Definite Loops

Alice

Repetition

- In many kinds of animations, especially simulations and games, some actions happen again and again.
  - Example: Gallery games where targets appear randomly on screen and then disappear only to appear elsewhere in the scene.
- Of course, actions are made to happen again and again by running an animation instruction (or a method) more than once

Example

- A bunny sneaks into a garden and wants to eat the broccoli. The bunny will need to hop several times to get to the broccoli.

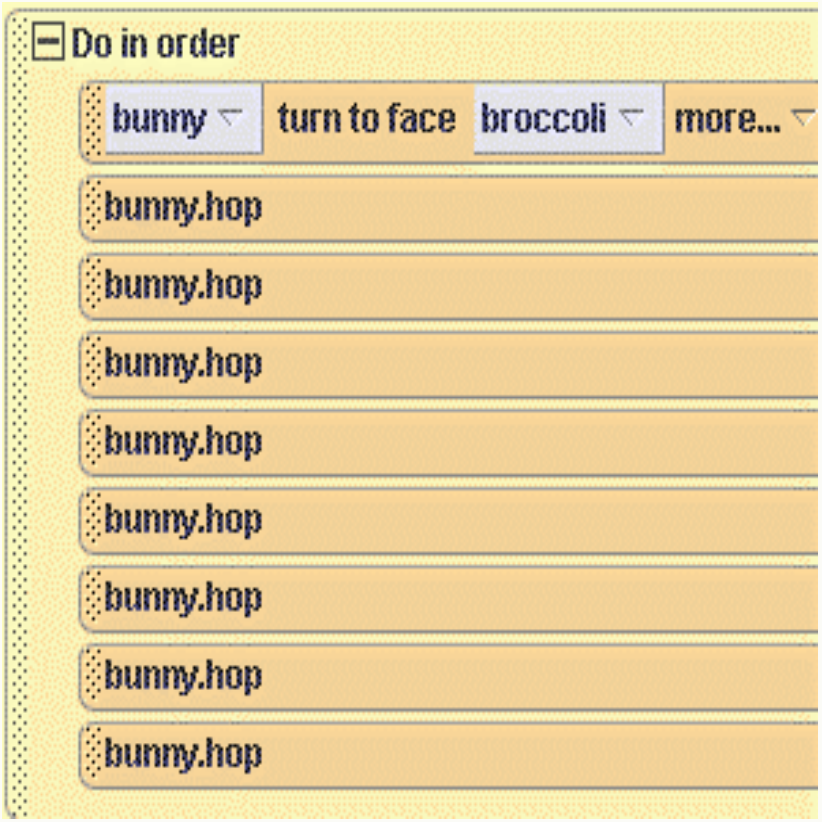
bunny.hop

bunny.hop *No parameters*

No variables

- Do together
  - bunny | play sound | World.jump (0:00.415) | *duration* = 0.5 seconds | more...
  - Do in order
    - // The Bunny moves
    - Do together
      - bunny | move | up | .5 meters | *duration* = 0.25 seconds | more...
      - bunny | move | forward | .4 meters | *duration* = 0.25 seconds | more...
    - Do together
      - bunny | move | down | 0.5 meters | *duration* = 0.25 seconds | more...
      - bunny | move | forward | 0.4 meters | *duration* = 0.25 seconds | more...
  - Do in order
    - // The Bunny's right foot simulates a hopping motion
    - bunny.hipR.footR | turn | forward | 0.12 revolutions | *duration* = 0.25 seconds
    - bunny.hipR.footR | turn | backward | 0.12 revolutions | *duration* = 0.25 seconds
  - Do in order
    - // The Bunny's left foot simulates a hopping motion
    - bunny.hipL.footL | turn | forward | 0.12 revolutions | *duration* = 0.25 seconds
    - bunny.hipL.footL | turn | backward | 0.12 revolutions | *duration* = 0.25 seconds

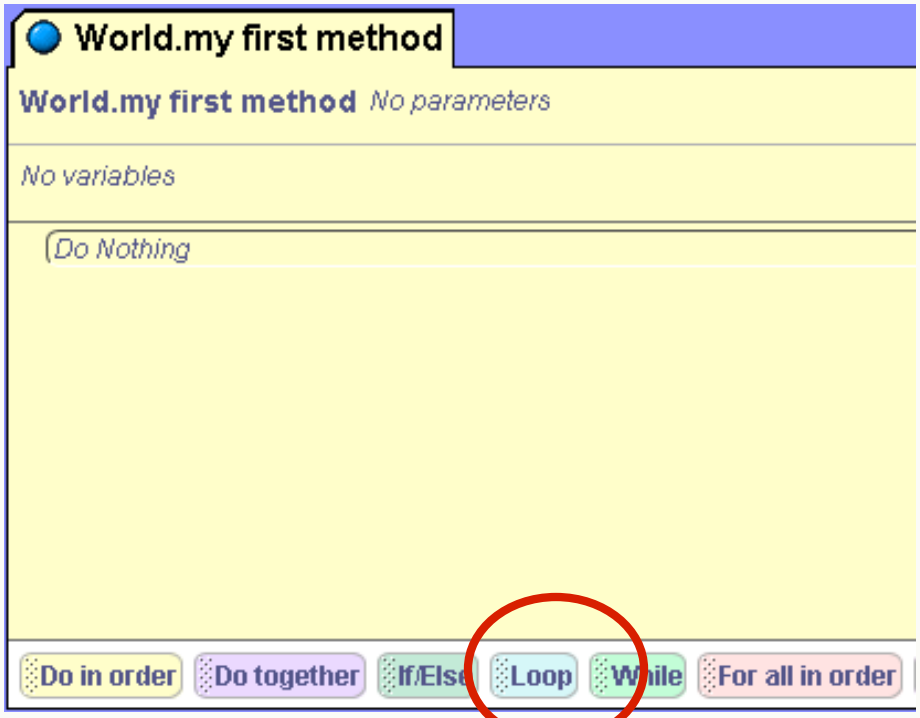
One solution



- Creating the same instruction again and again is somewhat tedious and the code gets longer and longer.

Counted Loop

- A counted loop is an alternate way to write repetitive code
  - Repeats instructions a counted number of times



Demo

- Ch07Lec1BunnyHop
- Concepts illustrated in this example
  - The loop instruction executes a definite number of times, specified by a count
  - Using a loop instruction
    - saves time
    - is convenient

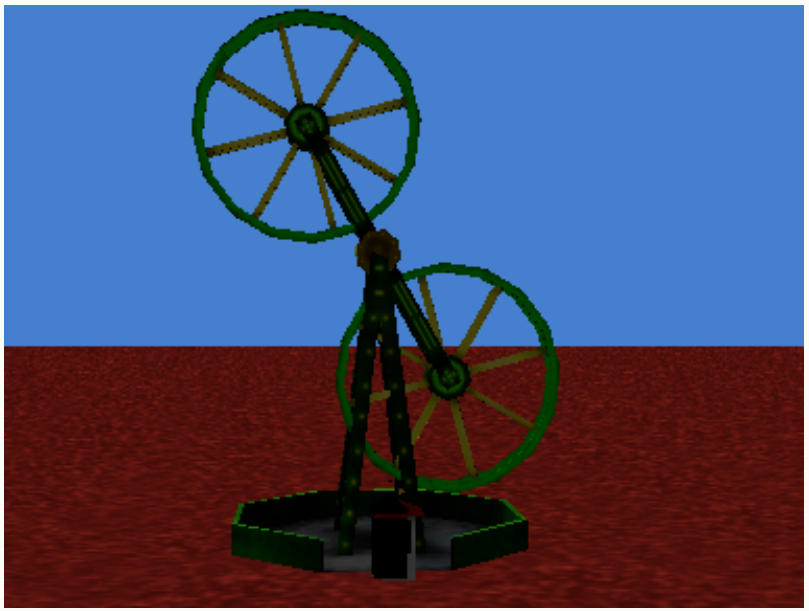
Demo

- Ch07Lec1Carouselinfinity
- Concept illustrated in this example
  - If “Infinity times” is selected for a loop, this means the loop will run until the program is shut down

More complicated loops

- It is also possible to place a loop statement within another loop statement
- This is called nested loops

An example of nested loops



The whole Ferris wheel will rotate clockwise, while the two inner wheels will rotate counterclockwise. The inner wheels should perform 2 revolutions for each outer loop revolution.

Demo

- Ch07Lec1FerrisWheel
- Concept illustrated in this example
  - The inner loop runs completely each time the outer loop runs once.
  - An outer loop that executes 2 times and an inner loop that executes 5 times will actually execute the inner loop 10 times.

Using a function

- A loop count can be computed by calling a function that returns a number value.
- The loop instruction automatically rounds the returned value to the nearest whole number.
- Demo: Ch07Lec1LoopWithFunctionCall

While: Indefinite Loops

Alice

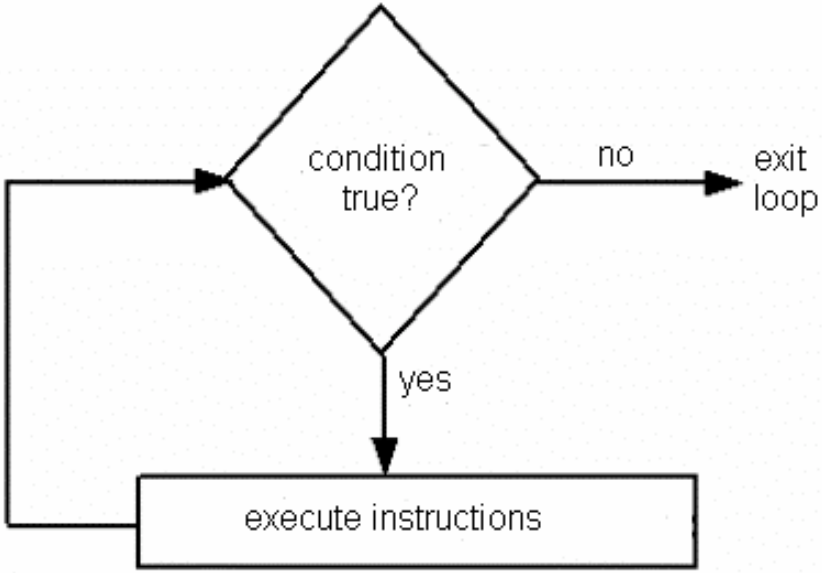
Repetition

- In some situations, we don't know exactly how many times a block of instructions should be repeated.
- All we know is that repetition is needed
  - For example, in a board game like chess or checkers, we don't know exactly how many moves it will take for a player to win or lose the game – all we know is that several moves will be needed.

Indefinite Repetition

- In programs where a count of repetitions is not known (indefinite), we can use one of two repetition control mechanisms:
  - *While* statement
  - Recursion
- This session focuses on the While statement.

How the While statement works



- The general idea is:

  While some condition is true
  execute instruction(s)

- To write a While statement, we need to know the condition that determines whether the loop will be repeated.

Example

- A common feature in popular "action films" is an exciting chase scene.
- As an illustration of an animated chase scene, consider the hungry shark in this world. The shark is going to chase after and catch a fleeing fish.

Problem

- The problem is how do we get the shark to chase the goldfish in a chase-like action?
  - The shark should not immediately catch the goldfish (otherwise, there would be no chase).
  - The goldfish (assuming self-preservation instincts) should appear to be fleeing.

Solution

- To create a chase scene,
  - At the same time, the shark will swim a short distance toward the fish and the fish will swim a short distance away from the shark.
  - The fish will flee to a random (but nearby) location.
  - As long as the goldfish is still 0.5 meters away from the shark, repeat the actions.

Storyboard

chase

While the goldfish is more than 0.5 meters away from the shark
    Do in order
        shark point at the goldfish
        Do together
            shark swim (toward the goldfish)
            goldfish flee (away from the shark)
shark eat (the goldfish)

The *shark swim, goldfish flee*, and *shark eat* actions are complex. Use stepwise refinement to break them down into simple steps.

chase

While the goldfish is more than 0.5 meters from the shark
    Do in order
        Point the shark at the goldfish
        Do together
            shark swim
            goldfish flee
shark eat (goldfish)

swim

tDo in order
    urn torso left and move forward
    turn torso right and move forward
    turn torso left and move forward

flee

Do together
  wiggle tail
  move to random location

Eat

Parameter: what
Do in order
    shark points at *what*
    shark opens jaw and *what* disappears
    shark closes jaw

Demo

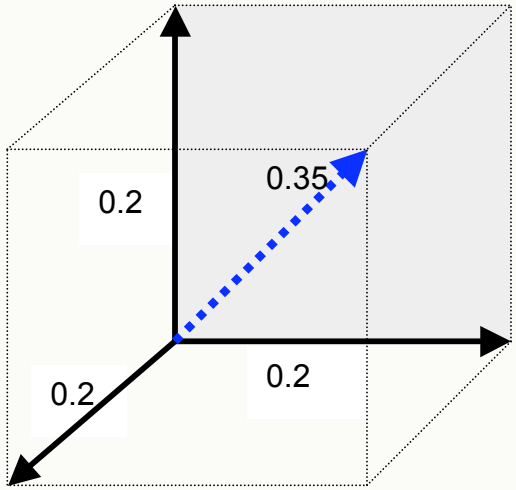
- Ch07Lec2Chase
- Concepts illustrated in this example
  - A While statement uses a Boolean condition to determine when the repetition ends.
  - Code written in a previous program can be reused in a new program.
    - In this example, the *flee* method calls the previously written *randomMotion* method.

Shark will catch goldfish

- How do we know the shark will eventually catch the goldfish?
  - The shark always moves 0.4 meters toward the goldfish
  - The goldfish's random motion is restricted by the *min* and *max* values used in the *random number* function.

The loop will end

- Geometrically, the fish can never move more than 0.35 meters away
- The shark has a distance advantage and will eventually catch up. The loop will end.



General “Rule of Thumb”

- As a general rule, a *While* loop should be written so the loop will eventually end.
  - Requires that statements within the loop change the conditions of the world such that the condition for the *While* statement will eventually become false.
- If the *While* loop never ends, it is an **infinite while** loop.